(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Patentschrift**

(10) **DE 43 15 732 C1**

(51) Int. Cl.[5]: G 06 F 12/14

(21) Aktenzeichen: P 43 15 732.7-53
(22) Anmeldetag: 11. 5. 93
(43) Offenlegungstag: —
(45) Veröffentlichungstag der Patenterteilung: 1. 6. 94

Innerhalb von 3 Monaten nach Veröffentlichung der Erteilung kann Einspruch erhoben werden

(73) Patentinhaber:

Siemens Nixdorf Informationssysteme AG, 33102 Paderborn, DE

(74) Vertreter:

Fuchs, F., Dr.-Ing., Pat.-Anw., 81541 München

(72) Erfinder:

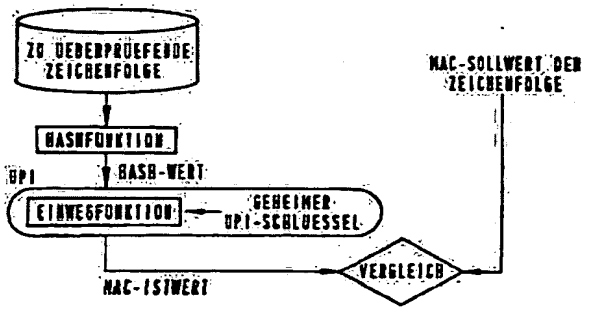
Osterlehner, Stefan, Dipl.-Ing., 8900 Augsburg, DE

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:

Groß, Michael, Vertrauenswürdiges Booten als Grundlage authentischer Basissysteme, in: Pfitzmann, A., Raubold E. (Hrsg.), VIS '91 - Verläßliche Informationssysteme, Proceedings (Informatik Fachberichte 271), Springer Verlag, März 1992, S. 190-207;
Bürk, Holger, Hashfunktionen in der Praxis der Datensicherung, in: DATASAFE, Tagungsband, Vde-Verlag Berlin, 1991, S. 189-198;
Beutelspacher, Albrecht: Rosenbaum, Ute; Kann man mit kryptographischen Methoden Viren erkennen? in: Paul, M. (Hrsg.), GI-19. Jahres- tagung I, Proceedings, Informatik-Fachberichte 222, Springer-Verlag, 1989, S. 564-577;
Hueske, Thomas;
Pfau, Axel;
Software-Versiegelung mit kryptographischen Methoden, in: Datenschutz und Datensicherung, Jg. 90, Nr. 6, Juni 1990, S. 298-304;

(54) Verfahren zum authentischen Booten und Testen der Integrität von Software auf PC-Architekturen

(57) Es wird ein Verfahren zum authentischen Booten und Testen von Software auf PC-Architekturen vorgeschlagen, welches ohne nennenswerte Komforteinbußen für den Benutzer die Integrität des Systems beim Booten überprüft. Als authentische, durch Software nicht manipulierbare Basiskomponente für das authentische Booten dient der in jedem PC vorhandene Ein-Chip-Mikrokontrollbaustein UPI. Dieser wird benutzt, um einen geheimen Schlüssel aufzunehmen und vor unautorisiertem Zugriff zu verbergen.



DE 43 15 732 C1

DE 43 15 732 C1

ISDOCID: <DE_ _4315732C1_I_>

Beschreibung

Die Erfindung betrifft ein Verfahren zum authentischen Booten und Testen der Integrität von Software auf PC-Architekturen.

Der Benutzer eines Informationstechnologie-Systems geht üblicherweise davon aus, daß die Funktionalität der benutzten Komponenten ihrer Spezifikation entspricht. Dabei vertraut er auf die Zuverlässigkeit der Hersteller von Hardware und Software. Ein Informationstechnologie-System eines vertrauenswürdigen Herstellers ist nur dann vertrauenswürdig, wenn die Hardware nicht manipuliert wurde und das System von einem authentischen Betriebssystem eines ebenfalls vertrauenswürdigen Herstellers gesteuert wird. Geht man von einer nicht manipulierten Hardware aus, so muß das Betriebssystem so gestartet werden, daß seine Authentizität vor dem Start überprüft wird.

Diese Vorgehensweise wird authentisches Booten genannt (vgl. Groß, Michael: "Vertrauenswürdiges Booten als Grundlage authentischer Basissysteme", in Pfitzmann A.; Raubold E.(Hrsg.): VIS '91-Verläßliche Informationssysteme", Proceedings (Informatik Fachberichte 271), S. 190—207, Springer, März 1992).

Authentisches Booten muß sich letztlich auf eine nicht manipulierbare Systemkomponente abstützen. Dies sollte ein Teil der Hardware sein, dessen Integrität gesichert ist. Nach dem authentischen Booten kann das Betriebssystem dazu benutzt werden, die Integrität von anderen Programmen zu überprüfen (z. B. mit kryptographisch abgesicherten Prüfsummen). Ein derartiges Vorgehen ist beispielsweise aus: Herda, Siegfried: "Chiffrier- und Signaturverfahren", Tutoriumsband "Verläßliche Informationssysteme", Deutsche Informatik Akademie, Bonn, 1991 bekannt.

Dieser integere Zustand bleibt häufig nicht lange bestehen. Wird beispielsweise ein neues Programm eingespielt und gestartet, dessen Herkunft unbekannt ist, so ist die Systemintegrität unter Umständen nicht mehr gewährleistet. Insbesondere bei PC's, deren hauptsächlich verwendete Betriebssysteme keinerlei Schutzmechanismen (MS-DOS) oder wenig Schutzmechanismen (OS/2) bieten, können beliebige Prozesse unautorisiert Änderungen an installierter Software vornehmen. Dies ist ein Hauptgrund, warum der Virenproblematik auf PC's so schwer wirkungsvoll zu begegnen ist. Viren können auf PC's nicht nur Anwendungsprogramme infizieren, sondern auch das Betriebssystem selbst manipulieren.

Dies führt dazu, daß einerseits Virenschutzmaßnahmen vereitelt werden können (z. B. durch als Tarnkappenviren bekannte Viren), andererseits die Virenausbreitung auch nach einem kompletten Rücksetzen (Reset) und Neustarten des Rechners u. U. weiter statt findet, selbst wenn danach kein virenverseuchtes Anwendungsprogramm gestartet wurde.

Tarnkappenviren installieren sich resident im Hauptspeicher und leiten die BIOS- und DOS-Funktionen zum Festplattenzugriff auf sich um. Liest ein Virenschutzprogramm eine Programmdatei oder den Boot-Sektor, um einen Virus zu finden, so "filtert" der Virus seinen Code beim Lesen selbst aus, so daß immer nur die Originaldateien gelesen werden und der Virus nicht bemerkt wird.

Gerade im Hinblick auf die Virenproblematik auf PC's ist also ein authentisches Booten notwendig, das Integritätsverletzungen meldet. Dadurch wird zwar eine Virenausbreitung nicht verhindert, aber diese immerhin dem Benutzer bekannt gemacht. Da die meisten Viren

Schadensfunktionen zeitverzögert ausführen, hat der Benutzer im allgemeinen die Möglichkeit, Gegenmaßnahmen zu treffen, bevor die Schadensfunktion ausgeführt wird.

Eine Möglichkeit, ein authentisches System zu booten, besteht bei PC's darin, das Betriebssystem immer von einer schreibgeschützten nicht manipulierten Original-Systemdiskette zu booten oder einen Betriebssystem-Integritätstest im ROM abzulegen. Beide Lösungen bringen Komfortverluste für den Anwender mit sich. Das Booten von Diskette dauert z. B. bei MS-DOS schon recht lange, bei OS/2 2.0 benötigt man zum Booten sogar zwei Disketten, mit denen der Systemstart etliche Minuten dauert. Die ROM-Lösung bringt Probleme bezüglich einer mangelnden Flexibilität gegenüber Versionsänderungen des Betriebssystems mit sich.

Aufgabe der Erfindung ist es, ein Verfahren zum authentischen Booten und Testen von Software auf PC-Architekturen anzugeben, durch das auf herkömmlichen PC-Architekturen ein authentisches Booten mit integriertem Software-Integritätstest möglich ist, ohne daß damit die genannten Nachteile verbunden sind.

Diese Aufgabe wird erfindungsgemäß durch ein Verfahren gelöst, das die Merkmale des Anspruchs 1 aufweist.

Bei diesem Verfahren wird eine kollisionsfreie Einweg-Hashfunktion angewendet, die in einem in einer PC-Architektur vorhandenen Universal Peripherie Schnittstellenbaustein (UPI) integriert ist.

Wie bereits erwähnt muß sich authentisches Booten letztlich auf einem integeren, durch Software nicht manipulierbaren Hardwarebaustein abstützen. Ein solcher Baustein kann eine geheime Information enthalten, auf die sich eine Kette von Integritätstests abstützt, die zum Start eines authentischen Betriebssystems führt.

Bei genauer Betrachtung der PC-Architektur findet man in jedem PC einen Baustein, der alle hier nötigen Eigenschaften aufweist — den UPI. Der UPI ist ein Schnittstellenbaustein, der einen kompletten Mikrocomputer (Prozessor, ROM, RAM, Taktgeber usw.) enthält. Er wird vorwiegend als intelligente Schnittstelle für die Tastatur und für die sichere Unterscheidung zwischen Kalt- und Warmstart verwendet. Im Programmspeicher des UPI ist genug Platz, um eine über einen geheimen Schlüssel (im folgenden UPI-Schlüssel genannt) parametrisierte Einwegfunktion unterzubringen, die als Ausgangspunkt einer Kette von Integritätsprüfungen dienen kann.

Vorteilhafte Ausgestaltungen der Erfindung sind Gegenstand von Unteransprüchen.

Danach erfolgt der Integritätstest in mehreren Stufen. Durch diese Maßnahme kann die hohe Flexibilität einer Disketten-Lösung und der Komfort und die Sicherheit einer ROM-Lösung miteinander kombiniert werden. Das Vorgehen in Stufen stellt sicher, daß kein Code ausgeführt wird, der nicht entweder unveränderbar im ROM liegt oder vor seiner Ausführung hinsichtlich seiner Integrität getestet wurde.

Um den Mißbrauch der Einwegfunktion durch ein nicht autorisiertes Programm (z. B. einen Virus) auszuschließen, kann der Zugriff auf diese Funktion nach ihrer Verwendung durch das autorisierte Programm abgeschaltet und erst bei einem Kaltstart wieder eingeschaltet werden. Die Kaltstart-Erkennung im UPI ist aufgrund der Hardware-Beschaltung zuverlässig und nicht umgehbar.

Wenn eine spezielle Speicherzelle (Security-Bit) im UPI programmiert wird, ist der Inhalt des UPI-Pro-

BNSDOCID: <DE__4315732C1_I_>

grammspeichers selbst nach dem Auslöten des Bausteins vor Auslesen geschützt (INTEL: "UPI — 41,42 — Universal Peripheral Interface 8-Bit Slave Microcontroller", Datenblatt, Intel Corporation, Santa Clara, CA, 1989, Seiten 9—54 bis 9—60, insbesondere Seite 9-60, Absatz: Security Bit). Damit könnte der UPI-Schlüssel in allen PC's identisch sein, ohne daß dadurch die Sicherheit eingeschränkt wird. Ein Ermitteln des UPI-Schlüssels durch eine Beobachtung der Kommunikation mit dem UPI ist bei einer geeigneten Auswahl der Einwegfunktion nicht möglich (siehe beispielsweise Simmons, Gustavus J.: "Contemporary Cryptology — The Science of Information Integrity", IEEE Press, Piscataway, NJ, 1992).

Das Verfahren weist insbesondere die Vorteile auf, daß keine Hardwareerweiterungen der PC-Architektur und auch keine Änderungen am Betriebssystem notwendig sind. Die oben beschriebenen Nachteile werden vermieden.

Kollisionsfreie Einweg-Hashfunktionen sind in: Bürk, Holger: "Hashfunktionen in der Praxis der Datensicherung", Beth, T.; Horster, P.: 'DATASAFE '91, Tagungsband', Seiten 189 bis 198, VDE-Verlag, Berlin, 1991 näher beschrieben.

Inwieweit man mit kryptographischen Methoden Viren erkennen kann ist von Beutelspacher, Albrecht; Rosenbaum, Ute: "Kann man mit kryptographischen Methoden Viren erkennen?", in Paul, M. (Hrsg.): 'GI — 19. Jahrestagung I', Proceedings, Informatik Fachberichte 222, Seiten 564 bis 577, Springer, 1989 beschrieben. Hieraus ist insbesondere der Begriff MAC (Message Authentication Code) bekannt, der in der weiteren Beschreibung verwendet wird.

Die Methode der Software-Versiegelung mit kryptographischen Methoden ist beispielsweise aus: Hueske, Thomas; Pfau, Axel: "Software-Versiegelung mit kryptographischen Methoden", Datenschutz und Datensicherung, Jg. 90, Nr. 6, Seiten 298 bis 304, Juni 1990 bekannt.

Nachfolgend wird ein Ausführungsbeispiel der Erfindung anhand einer Zeichnung näher erläutert. Es zeigen

Fig. 1 ein Struktogramm mit einem Ablauf des Systemstarts mit Integritätstest gemäß der Erfindung,

Fig. 2 einen ersten Integritätstest mit UPI beim Ablauf nach Fig. 1,

Fig. 3 einen zweiten Integritätstest einer Security-Partition beim Ablauf nach Fig. 1,

Fig. 4 einen Lageplan mit Interaktionspfeilen von einzelnen beteiligten Komponenten beim Ablauf nach Fig. 1.

Als Grundlage für die Beschreibung wird eine PC-Architektur mit einem allgemein bekannten Betriebssystem wie zum Beispiel das MS-DOS- oder das OS/2-Betriebssystem angenommen. PC-Architekturen weisen einen Prozessor auf, der alle nötigen Eigenschaften einer Chipkarte aufweist. Mit einer Chipkarte kann verhindert werden, daß eine Einwegfunktion von einem Virus mißbraucht werden kann. Mit einer Chipkarte kann ein Hash-Wert beim Anwender verschlüsselt und bei jedem Test entschlüsselt werden. Der Nachteil der Chipkarte oder auch einer Magnetkarte oder einer Diskette, beispielsweise ist, daß der Datenträger mit der Einwegfunktion nur bei Ablauf des Prüfprogramms eingesteckt sein darf und nach dem Test zuverlässig entnommen werden muß, sonst könnte der Virus auch von der Einwegfunktion Gebrauch machen.

Bei dem Prozessor handelt es sich um einen Universal Peripherie Schnittstellenbaustein (Universal Peripheral Interface UPI), der vorwiegend als intelligente Schnittstelle für die Tastatur und für die sichere Unterscheidung zwischen einem Kalt- und einem Warmstart verwendet wird.

Im Programmspeicher des UPI ist genug Platz, um eine Einwegfunktion unterzubringen. Um den Mißbrauch durch einen Virus auszuschließen, kann der Zugriff auf diese Funktion nach ihrer Verwendung abgeschaltet und erst bei einem Kaltstart wieder eingeschaltet werden. Wenn eine spezielle Speicherzelle (Security-Bit) im UPI programmiert wird, ist selbst nach dem Auslöten des Bausteins ein Auslesen des UPI-Programms nicht mehr möglich. Damit kann die Einwegfunktion in allen PC-Architekturen identisch sein ohne daß dadurch die Sicherheit eingeschränkt wird. Durch die sichere Abschaltbarkeit des Algorithmus im UPI kann kein Vergessen einer Chipkarte oder Diskette Probleme mehr bereiten.

Es wäre denkbar, daß bei immer gleicher Verschlüsselung im UPI ein Angreifer auf einem anderen PC eine Signatur zu einer manipulierten Hash-Datei berechnet und dann bei der Vireninfektion überträgt. Aus diesem Grund wird bei der Hash-Wert-Ermittlung eine individuelle (z. B. von den Konfigurationsmerkmalen des PC abhängige) Seriennummer als ein Hash-Startwert verwendet. Dadurch sind die Signaturen auch bei gleichen Programmen auf jedem PC verschieden.

Die Verschlüsselung des Hash-Wertes kann auch mit seiner Ermittlung zusammengelegt werden. Dadurch kann auf eine nachträgliche Verschlüsselung verzichtet werden und die UPI-Einwegfunktion nur zum Erzeugen eines "geheimen Schlüssels" verwendet werden. Dieser Schlüssel kann dann noch verwendet werden, nachdem die UPI-Funktion abgeschaltet worden ist. Dabei wird nicht der Hash-Wert verschlüsselt sondern ein "geheimer Schlüssel" — der durch Verschlüsselung einer Seriennummer gebildet wurde — als Hash-Startwert verwendet.

Als Grundlage für die weitere Beschreibung wird eine PC-Architektur mit einem allgemein bekannten Betriebssystem wie zum Beispiel das MS-DOS- oder das OS/2-Betriebssystem angenommen.

Bevor der Ablauf beim Booten näher erläutert wird, werden die Maßnahmen beschrieben, die notwendig sind, um das Schutzsystem in der PC-Architektur zu installieren. Dazu sind in einem Basis-Ein-Ausgabe-System in einem Festwertspeicher (Read Only Memory-Basic Input Output System ROM-BIOS) das BIOS und das UPI-Programm zu erweitern und eine Spezial-Partition (Security-Partition) sowie ein Testprogramm als Gerätetreiber neu zu erstellen. Eine Partition ist ein Einteilungsbereich einer Festplatte, innerhalb dem der Code von Software gespeichert ist.

Das BIOS muß um folgende Funktionen erweitert werden:

— Test (und eventuell Neuermittlung und Abspeicherung der Signatur einer Urladers (Partition Loader),
— Test (und eventuell Neuermittlung und Abspeicherung der Signatur) der Spezial-Partition,
— Nutzung der Verschlüsselungsfunktion (Einwegfunktion) im UPI,
— Abschaltung der Verschlüsselungsfunktion im UPI,
— Ablage eines "geheimen Schlüssels" im Hauptspeicher, damit auch nach Abschaltung der UPI-Verschlüsselungsfunktion noch (von der Spezial-

SDOCID: <DE_ _4315732C1_I_>

Partition aus) eine symmetrische Verschlüsselung durchgeführt werden kann.

Das UPI-Programm muß um eine Verschlüsselungsfunktion erweitert werden, die nur zwischen Kaltstart und Abschaltung durch das BIOS arbeitet. Dazu sind folgende Programmteile zu implementieren:

— Verschlüsselungsalgorithmus (Einwegfunktion), der nur arbeitet, wenn ein bestimmtes Byte in einem UPI-RAM gesetzt ist,
— Verwaltung eines Bytes im UPI-RAM (Setzen beim Kaltstart (Reset), Rücksetzen auf BIOS-Befehl,
— Funktion zum Datentransfer zwischen UPI und Hauptprozessor,
— Änderungen an bereits vorhandenen Funktionen, mit denen das Byte um UPI-RAM rücksetzbar wäre.

Das eigentliche Testprogramm liegt auf einer extra Partition, der Security-Partition. Diese Partition kann keine primäre (bootfähige) DOS- oder OS/2-Partition sein, da MS-DOS und OS/2 nur eine primäre DOS- oder OS/2-Partition erlauben und diese selbst benötigen. Aus diesem Grund muß ein eigenes einfaches Dateisystem und ein eigenes kleines Betriebssystem installiert werden.

Für die Spezial-Partition sind folgende Teile nötig:

— Minimal-Betriebssystem mit Programm-Laderoutine,
— Dateisystem-Treiber (für die zu prüfenden Partitionen),
— Prüfung der eigenen Dateien (mit Hilfe des "geheimen Schlüssels")
— (Optionaler Boot-Manager, mit dem die Partition ausgewählt wird, von der das Betriebssystem gestartet wird),
— Prüfung der veränderlichen Daten des zu testenden Betriebssystems, zum Beispiel Boot-Sektor und zugehörige Konfigurations-Datei CONFIG.SYS (mit Hilfe des "geheimen Schlüssels"),
— ein sogenannter RSA-Signaturtest (RSA steht für das Rivest-Shamier-Adleman-Verfahren, siehe Groß, Michael, "Vertrauenswürdiges Booten ...," Seite 194) einer Hash-Datei C: /SECUITY-DAT,
— Prüfung von Betriebssystem und Programmen (alle Dateien , die in SECURITY.DAT mit ihren Hash-Werten verzeichnet sind) und für jede EXE-Datei ein Test, ob eine neue gleichnamige COM-Datei existiert,
— Löschen des "geheimen Schlüssels" aus dem Hauptspeicher,
— Starten des Betriebssystems (MS-DOS oder OS/2).

Sind in dem zu testenden System weitere Datenträger, die nur mit einem Gerätetreiber (in CONFIG.SYS) ansprechbar sind (z. B. eine High Performance File System-Partition HPFS-Partition), so kann nach dem Laden des Betriebssystems und der Treiber ein weiteres Prüfprogramm gestartet werden. Dieses Prüfprogramm kann von CONFIG.SYS aus in Form eines Gerätetreibers oder als normale Programm-Datei gestartet werden. Durch die vorher ausgeführten Prüfstufen ist sichergestellt, daß Betriebssystem, CONFIG.SYS und das Prüfprogramm integer sind.

Der Ablauf beim Booten erfolgt in mehreren Schritten:

1. Ein im ROM-BIOS integrierter Integritätstest prüft nach dem Kaltstart die Integrität des Partitions-Laders und Teile einer (zum Konzept gehörenden) Spezial-Partition der Festplatte.
2. Das Prüfprogramm auf der Spezial-Partition testet, vor dem Start des Betriebssystems, die Integrität von Boot-Sektor, Betriebssystem und anderen wichtigen ausführbaren Dateien.
3. Ein weiteres Prüfprogramm testet nach dem Start des Betriebssystems (z. B. als über die Datei CONFIG.SYS in das Betriebssystem eingebundener Gerätetreiber) die Integrität von Dateien auf weiteren Datenträgern, die nur mit Hilfe des Betriebssystems oder Gerätetreibern ansprechbar sind (z. B. HPFS-Partitionen bei OS/2).

Fig. 1 zeigt den Ablauf beim Booten einschließlich des Software-Integritätstests in Form eines Struktogramms. Die gegenüber dem normalen Bootvorgang neu hinzugekommenen Teile sind strichpunktiert eingerahmt. Die Funktionen, die durch Programme auf der Security-Partition realisiert werden, sind mit einem gestrichelten Rahmen gekennzeichnet Nachfolgend wird der Ablauf im einzelnen erläutert:

Nach dem Kaltstart initialisiert und testet der POST (Power On Self Test) des BIOS die Hardware. Anschließend werden Optionen-ROM's (z. B. mit SCSI-Treiber gesucht und eingebunden.

Nun kommt die BIOS-Erweiterung zum Integritätstest des Partitions-Laders und der Spezial-Partition zur Ausführung. Zunächst wird mit Hilfe des UPI überprüft, ob ein Kaltstart vorliegt. Das ist nötig, da die später benötigte Einweg-Funktion im UPI nach einem Kaltstart aktiv ist. Es wird nun der Partitions-Lader gelesen und mit Hilfe einer kryptographischen Hash-Funktion ein eindeutiges Kondensat (Hash-Wert) abgeleitet. Dieser Hash-Wert wird durch den UPI verschlüsselt und mit dem Sollwert verglichen, der an einer bestimmten Stelle auf der Security-Partition abgespeichert ist. Diese Werte werden mit MAC (Message Authentication Code) bezeichnet. Auf die gleiche Weise wird das Programm auf der Security-Partition getestet. Die dafür nötigen Daten (MAC-Werte und Länge des zu testenden Security-Programms) sind im ersten Sektor (Boot-Sektor) der Security-Partition gespeichert.

Fig. 2 zeigt den mit Hilfe des UPI ablaufenden ersten Integritätstest in graphischer Form.

Stimmt der MAC-Istwert mit dem gespeicherten MAC-Sollwert nicht überein (z. B. beim ersten Programmlauf nach der Installation oder bei Virenbefall) so erhält der Anwender eine Meldung. Beim ersten Programmlauf kann der Anwender (z. B. nach Angabe eines Paßwortes) die neu ermittelten MAC-Werte authentifizieren und dann abspeichern lassen. Treten bei einem späteren Programmlauf Differenzen auf, so liegt eine Integritätsverletzung (z. B. Virenbefall) vor. In diesem Fall können die veränderten Teile von einer Sicherungsdiskette restauriert werden.

Liegt keine Diskette in Laufwerk A, wird eine Seriennummer (die z. B. aus der Hardware-Konfiguration des PC's abgeleitet wird) im UPI verschlüsselt und als zu diesem PC gehöriger geheimer Schlüssel im Hauptspeicher abgelegt. Dann wird die Verschlüsselungsfunktion des UPI abgeschaltet. Sie wird erst wieder durch einen Kaltstart hardwaremäßig eingeschaltet.

BNSDOCID: <DE___4315732C1_I_>

Jetzt lädt wie gewohnt das BIOS den standardmäßigen Partitions-Lader von der Festplatte und startet seinen Code. Der Partitions-Lader wiederum lädt und startet den Boot-Sektor der Security-Partition. Der zweite Integritätstest beginnt. Der Boot-Sektor lädt und startet das Security-Programm von der Security-Partition. Das Security-Programm ermittelt wieder mit einer Hash-Funktion den Hash-Wert der Daten auf der Security-Partition. Dieser Wert wird dann mit einer durch den vom BIOS abgelegten geheimen Schlüssel parametrisierten Einwegfunktion verschlüsselt und mit dem ebenfalls in der Security-Partition abgespeicherten MAC-Sollwert verglichen (siehe Fig. 3). Damit wurde die Integrität der Security-Partition überprüft.

Die Daten auf der Security-Partition umfassen z. B. die MAC-Werte von veränderbaren Betriebssystem-Teilen (Boot-Sektor und CONFIG.SYS), die Pfadangaben von Hash-Dateien verschiedener Programmpakete, die vom Hersteller mitgeliefert werden sollten, sowie Hash-Werte von Programmen, die ohne Signaturdatei geliefert wurden.

Der geheime Schlüssel im Hauptspeicher wird nun nicht mehr benötigt und daher gelöscht. Die weiteren auszuführenden Integritätstests arbeiten mit asymmetrischen kryptographischen Verfahren.

Auf der zu prüfenden Betriebssystem-Partition befindet sich die Hash-Datei C:/SECURITY.DAT, die Namen, Pfad und Hash-Werte aller zu testenden Programmdateien enthält. Sie wird vom Betriebssystemhersteller (oder -händler) erstellt, mit einer Signatur versehen und mitgeliefert. Die Signaturen werden mit einem asymmetrischen kryptographischen Verfahren (z. B. RSA) erzeugt und verifiziert. Der zur Verifikation der Signatur der Datei notwendige öffentliche Schlüssel ist auf der bereits getesteten Security-Partition abgelegt und daher nicht manipuliert worden. Ergibt diese Überprüfung der Signatur, daß die Datei authentisch ist, so werden alle darin angegebenen Programmdateien überprüft, indem ihr Hash-Wert ermittelt und mit dem in SECURITY.DAT verzeichneten verglichen wird. Durch diese Vorgehensweise wird eine sichere Software-Distribution zwischen Hersteller und Kunden erreicht. Auf die selbe Weise wird mit weiteren Hash-Dateien verfahren, deren Namen und Lage nach der Installation von neuen Programmpaketen in die Security-Daten aufgenommen wurden. Nach Abschluß dieser Prüfung wird der Boot-Sektor des Betriebssystems geladen und gestartet und damit das Betriebssystem gebootet.

In der Konfigurationsdatei CONFIG.SYS werden Gerätetreiber geladen. Hier kann auch ein weiteres Prüfprogramm angegeben sein, das dann (ähnlich wie das Programm auf der Security-Partition) Dateien auf anderen Datenträgern, die nur mit Gerätetreibern ansprechbar sind (z. B. HPFS-Partitionen) überprüft. Jetzt wird die Benutzeroberfläche wie gewohnt geladen. Alle Integritätsprüfungen sind abgeschlossen und behindern den weiteren Rechnerbetrieb nicht.

Eine Abwandlung gegenüber den Ausführungsformen gemäß der Fig. 2 und 3 besteht darin, mit der Einwegfunktion im UPI aus einer gerätespezifischen Seriennummer einen "geheimen Schlüssel" zur Parametrisierung zu bilden und mit diesem über die Hash-Funktion den Hash-Wert zu ermitteln.

Eine weitere Ausführungsform ist, jeweils die Form gemäß der Fig. 2 und 3 mit der oben beschriebenen Möglichkeit zu kombinieren, das heißt gleichzeitig beide Möglichkeiten vorzusehen und dann eine jeweils gewünschte Möglichkeit zu aktivieren.

Fig. 4 zeigt noch einmal die Lage und Interaktion der einzelnen beteiligten Komponenten. Die durchgezogenen Pfeile stehen dabei für ein Laden, die gestrichelten Pfeile stehen für ein Prüfen und die gepunkteten Pfeile stehen für ein Daten lesen/schreiben.

Die Bedeutungen der einzelnen mit Buchstaben bezeichneten Kästchen ist folgende:
A-Code für Test des Partition Loaders,
B-Code für Test der Security Partition,
C-Code zur Berechnung der Signaturen,
D-Verwaltung eines Abschalt-Bytes,
E-Einwegfunktion,
F-"geheimer Schlüssel",
G-Unveränderter Partition-Loader-Code,
H-Partition-Tabelle: Security Partition,
I-Code-Größe,
J-Code-Signatur; Partition-Loader-Signatur,
K-Boot-Sektor; Loader; DOS-Emulator;
L-Prüfgrogramm und Betriebssystem-Booter,
M-Signatur-Daten,
N-Betriebssystem-Boot-Sektor,
O-Betriebssystem-Loader, Kernel, Treiber,
P-CONFIG.SYS,
Q-Signatur-Datei des Betriebssystems,
R-weiteres Prüfprogramm,
S-Programme und Daten,
T-Signatur-Dateien.

Patentansprüche

1. Verfahren zum authentischen Booten und Testen der Integrität von Software auf PC-Architekturen mit einem universalen Peripherie Schnittstellenbaustein (Universal Peripheral Interface UPI), unter Verwendung einer kryptographischen Siegelerzeugungs- und -prüfungsmethode gemäß einem symmetrischen *Versiegelungsverfahren* für eine Versiegelung der Software sowie unter Verwendung eines Bootvorganges, bei dem folgende Schritte ablaufen:
   - — Durchführen eines Einschalt-Selbst-Testes (Power On Self Test, POST) in einem Basis-Ein-Ausgabe-System in einem Festwertspeicher (Read Only Memory-Basic Input Output System, ROM-BIOS)
   - — Laden und Ausführen eines Urladers (Partition-Loader) für einen Code eines Einteilungsbereiches (Partition) einer Festplatte, innerhalb dem Software gespeichert ist,
   - — Laden und Ausführen eines Boot-Sektors einer Betiebssystem-Partition,
   - — Laden und Ausführen eines Laders (Loader) zum Laden und Starten eines Betriebssystems, und
   - — Laden und Starten eines Betriebssystemkerns (Kernel), wobei für die Siegelerzeugung und -prüfung eine im UPI fest eingebundene kollisionsfreie Einweg-Hashfunktion angewendet wird.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet,
daß durch einen im ROM-BIOS integrierten ersten Integritätstester nach einem vom UPI erkannten Kaltstart und dem POST des ROM-BIOS die Integrität des Urladers und einer weiteren Sicherheits-Partition (Security Partition) mit einem zweiten Integritätstester auf der Festplatte unter Verwendung der im vorhandenen UPI integrierten Ein-

ISDOCID: <DE_ _4315732C1_I_>

weg-Hashfunktion geprüft und beim Booten von der Festplatte ein auf die Einweg-Hashfunktion aufbauender geheimer kryptographischer Schlüssel in einem Schreib-Lese-Speicher (RAM) abgelegt und dann die Einweg-Hashfunktion abgeschaltet wird, und

daß beim Kaltstart mit Booten von der Festplatte vor dem Start des Betriebssystems durch den zweiten Integritätstester unter wenigstens zeitweiser Zuhilfenahme des im RAM abgelegten geheimen kryptographischen Schlüssels die Integrität der Daten der Sicherheits-Partition, des Boot-Sektors des Betriebssystems, des Betriebssystems und sonstiger wichtiger ausführbarer Dateien geprüft und bei nicht mehr Benötigen des geheimen Schlüssels bei dieser Prüfung der geheime Schlüssel gelöscht wird.

3. Verfahren nach Anspruch 2, dadurch gekennzeichnet, daß nach dem Start des Betriebssystems durch einen weiteren Integritätstester die Integrität von Dateien auf weiteren Datenträgern, die nur mit Hilfe des Betriebssystems oder zugehörigen Gerätetreibern ansprechbar sind, mit kryptographischen Methoden geprüft wird.

4. Verfahren nach Anspruch 2 oder 3, dadurch gekennzeichnet, daß beim Aktivieren des zweiten Integritätstesters als erstes ein Minimalbetriebssystem geladen und gestartet wird.

5. Verfahren nach Anspruch 1 bis 4, dadurch gekennzeichnet, daß bei einer Hashfunktion zur Ableitung eines Hashwertes bei der Siegelerzeugung und -prüfung zum Zweck der Parametrisierung eine auf eine PC-Architektur bezogene Seriennummer berücksichtigt wird.

6. Verfahren nach Anspruch 5, dadurch gekennzeichnet, daß aus der Seriennummer in Verbindung mit der Einweg-Hashfunktion im UPI ein "geheimer Schlüssel" erzeugt wird, mit dem über die Hashfunktion ein Hash-Wert abgeleitet wird.

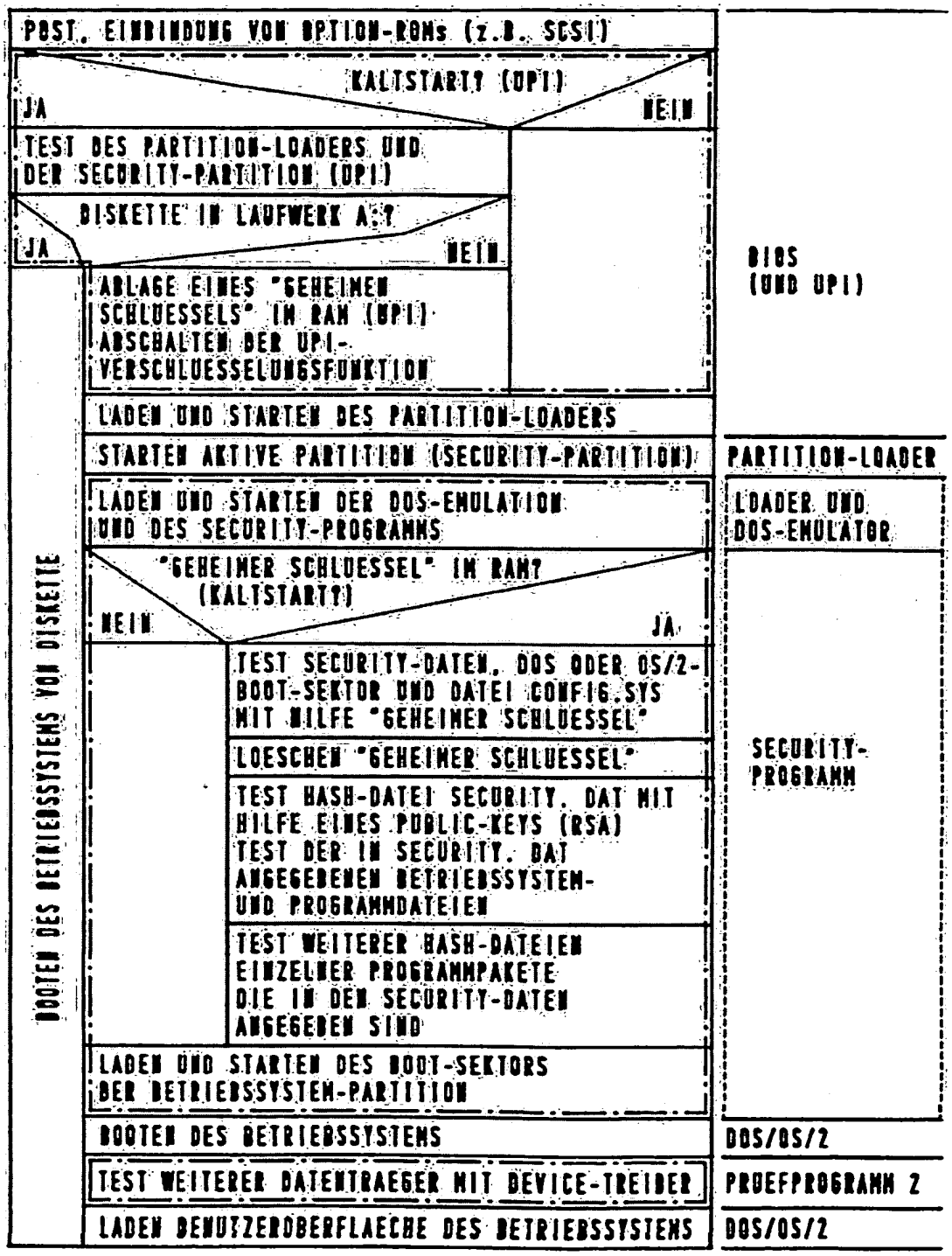
7. Verfahren nach einem der vorherigen Ansprüche, dadurch gekennzeichnet, daß bei der Durchführung der Einweg-Hashfunktion ein im UPI integrierter geheimer Schlüssel berücksichtigt wird.

Hierzu 3 Seite(n) Zeichnungen

BNSDOCID: <DE___4315732C1_I_>

Nummer: DE 43 15 732 C1
Int. Cl.[5]: G 06 F 12/14
Veröffentlichungstag: 1. Juni 1994

FIG 1



408 122/370

BNSDOCID: <DE___4315732C1_I. >

Nummer: DE 43 15 732 C1
Int. Cl.[5]: G 06 F 12/14
Veröffentlichungstag: 1. Juni 1994

FIG 2

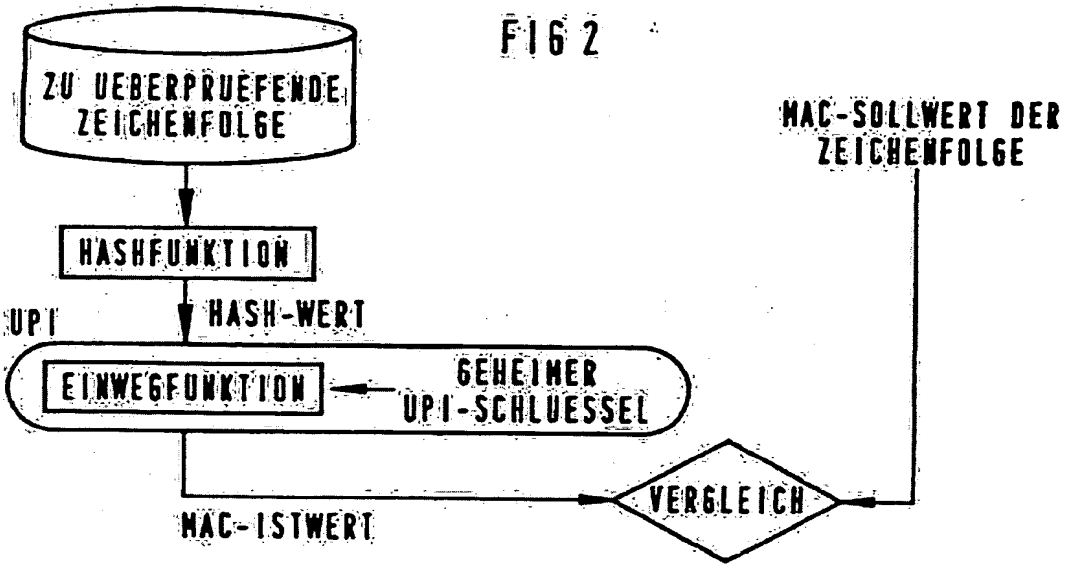
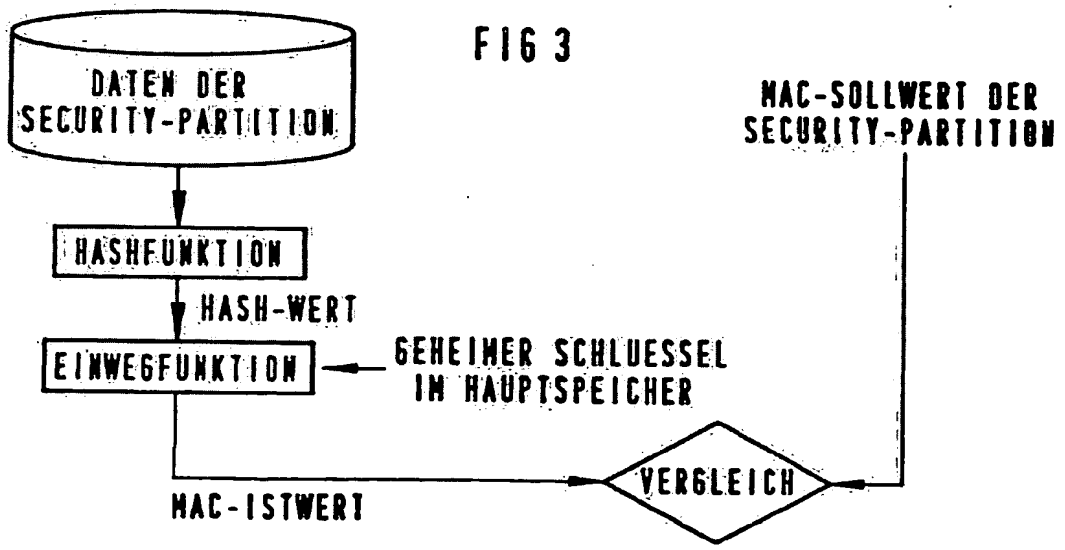


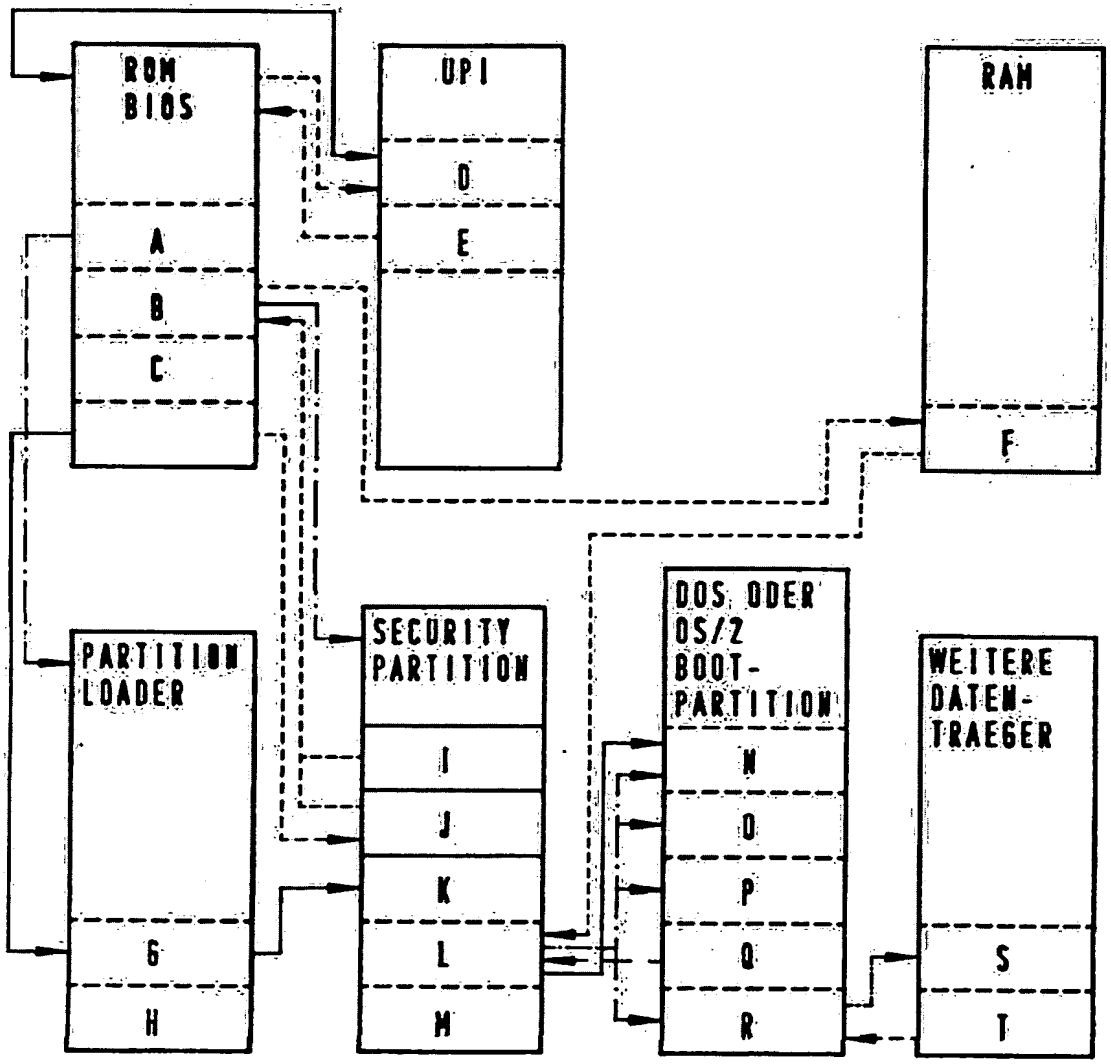
FIG 3



ISDOCID: <DE_ _4315732C1_I_>

Nummer: DE 43 15 732 C1
Int. Cl.[5]: G 06 F 12/14
Veröffentlichungstag: 1. Juni 1994

FIG 4



3NSDOCID: <DE __4315732C1_I_>